Experience in Using SIMD and MIMD Parallelism for Computational Fluid Dynamics

Horst D. Simon[1] and Leonardo Dagum[1]
Applied Research Branch
Numerical Aerodynamic Simulation (NAS) Systems Division
NASA Ames Research Center, Mail Stop T045-1
Moffett Field, CA 94035

March 21, 1991

Abstract

One of the key objectives of the Applied Research Branch in the Numerical Aerodynamic Simulation (NAS) Systems Division at NASA Ames Research Center is the accelerated introduction of highly parallel machines into a full operational environment. In this report we summarize some of the experiences with the parallel testbed machines at the NAS Applied Research Branch. We discuss the performance results obtained from the implementation of two Computational Fluid Dynamics (CFD) applications, an unstructured grid solver and a particle simulation, on the Connection Machine CM-2 and the Intel iPSC/860.

Keywords: parallel architectures, MIMD, SIMD, computational fluid dynamics.
AMS Subject Classification 76-08, 65W05, 65N99.
CR Subject Classification G.1.8, J.2, C.1.1, C.1.2.

[1]The author is an employee of Computer Sciences Corporation. This work is supported through NASA Contract NAS 2-12961

1 Introduction

One of the key tasks of the Applied Research Branch in the Numerical Aerodynamic Simulation (NAS) Systems Division at NASA Ames Research Center is the accelerated introduction of highly parallel and related key hardware and software technologies into a full operational environment (see [1]). From 1988 - 1990 a testbed facility has been established for the development and demonstration of highly parallel computer technologies. Currently a 32k processor Connection Machine CM-2 and an 128 node Intel iPSC/860 are operated at the NAS Applied Research Branch. This testbed facility is envisioned to consist of successive generations of increasingly powerful highly parallel systems that are scalable to high performance capabilities beyond that of conventional supercomputers.

It is recognized within the scientific computing community that the most promising approach toward achieving very large improvements in computing performance is through the application of highly parallel architectures. To meet the future processing needs of the aerospace research community, the Applied Research Branch supports a research program aimed at achieving the best match of parallel processing technology to the most demanding research applications. In the last two years a number of large scale computational fluid dynamics applications have been implemented on the two testbed machines, and the potential of the parallel machines for production use has been evaluated. Beyond that, a systematic performance evaluation effort has been initiated (see [4]), and basic algorithm research has been continued.

In this report we will first give a brief description of the capabilities of the parallel machines at NASA Ames. Then we will discuss some of the research carried out in the implementation of Computational Fluid Dynamics (CFD) applications on these parallel machines. We focus here on those applications where we have more detailed knowledge because of our own involvement: an explicit 2D Euler solver for unstructured grids, and a simulation based on particle methods. Other applications based on structured grids will be mentioned briefly, as well as the NAS effort in parallel benchmarking. In a final section we offer some preliminary conclusions on the performance of current parallel machines for CFD applications, as well as the potential of the different architectures for production use in the future. Another summary of some of the results from NASA Ames is given by D. Bailey in [3].

2 Parallel Machines at NASA Ames

2.1 Connection Machine

The Thinking Machines Connection Machine Model CM-2 is a massively parallel SIMD computer consisting of many thousands of bit serial data processors under the direction of a front end computer. The system at NASA Ames consists of 32768 bit serial processors each with with 1 Mbit of memory and operating at 7 MHz. The processors and memory are packaged as 16 in a chip. Each chip also contains the routing circuitry which allows any processor to send and receive messages from any other processor in the system. In addition, there are 1024 64-bit Weitek floating point processors which are fed from the bit serial processors through a special purpose "Sprint" chip. There is one Sprint chip connecting every two CM chips to a Weitek. Each Weitek processor can execute an add and a multiply each clock cycle thus performing at 14 MFLOPS and yielding a peak aggregate performance of 14 GFLOPS for the system.

The Connection Machine can be viewed two ways, either as an 11-dimensional hypercube connecting the 2048 CM chips or a 10-dimensional hypercube connecting the 1024 processing elements. The first view is the "fieldwise" model of the machine which has existed since its introduction. This view admits to the existence of at least 32768 physical processors (when using the whole machine) each storing data in fields within its local memory. The second is the more recent "slicewise" model of the machine which admits to only 1024 processing elements (when using the whole machine) each storing data in slices of 32 bits distributed across the 32 physical processors in the processing element. Both models allow for "virtual processing", where the resources of a single processor or processing element may be divided to allow a greater number of virtual processors.

Regardless of the machine model, the architecture allows interprocessor communication to proceed in three manners. For very general communication with no regular pattern, the router determines the destination of messages at run time and directs the messages accordingly. This is referred to as general router communication. For communication with an irregular but static pattern, the message paths may be pre-compiled and the router will direct messages according to the pre-compiled paths. This is referred to as compiled communication and can be 5 times faster than general router communication.

Finally, for communication which is perfectly regular and involves only shifts along grid axes, the system software optimizes the data layout by ensuring strictly nearest neighbor communication and uses its own pre-compiled paths. This is referred to as NEWS (for "NorthEastWestSouth") communication. Despite the name, NEWS communication is not restricted to 2-dimensional grids, and up to 31-dimensional NEWS grids may be specified. NEWS communication is the fastest.

The I/O subsystems connect to the data processors through an I/O controller. An I/O controller connects to 8192 processors through 256 I/O lines. There is one line for each chip but the controller can only connect to 256 lines simultaneously and must treat its 8k processors as two banks of 4k each. Each I/O controller allows transfer rates of up to 40 MB per second. In addition to an I/O controller there can be a frame buffer for color graphics output. Because it is connected directly to the backplane rather than through the I/O bus, the frame buffer can receive data from the CM processors at 256 MB per second. The system at NASA Ames has two frame buffers connected to two high resolution color monitors and four I/O controllers connected to a 20 GB DataVault mass storage system.

The Connection Machine's processors are used only to store data. The program instructions are stored on a front end computer which also carries out any scalar computations. Instructions are sequenced from the front end to the CM through one or more sequencers. Each sequencer broadcasts instructions to 8192 processors and can execute either independent of other sequencers or combined in two or four. There are two front end computers at NASA Ames, a Vax 8350 and a Sun 4/490, which currently support about 100 users. There are two sequencer interfaces on each computer which allow up to four concurrent users. In addition, the system software supports the Network Queue System (NQS) and time sharing through the CM Time Sharing System (CMTSS).

The Connection Machine system was first installed at NASA Ames in June of 1988. Since then the system has undergone a number of upgrades, the most recent being completed in February of 1991. An assessment of the system is given in [25]. Perhaps its greatest strength, from a user standpoint, is the robust system software. This is of critical importance to NASA as it moves its parallel machines into production mode.

2.2 Intel iPSC/860

The Intel iPSC/860 (also known as Touchstone Gamma System) is based on the new 64 bit i860 microprocessor by Intel [14]. The i860 has over 1 million transistors and runs at 40 MHz. The theoretical peak speed is 80 MFLOPS in 32 bit floating point and 60 MFLOPS for 64 bit floating point operations. The i860 features 32 integer address registers, with 32 bits each, and 16 floating point registers with 64 bits each (or 32 floating point registers with 32 bits each). It also features an 8 kilobyte on-chip data cache and a 4 kilobyte instruction cache. There is a 128 bit data path between cache and registers. There is a 64 bit data path between main memory and registers.

The i860 has a number of advanced features to facilitate high execution rates. First of all, a number of important operations, including floating point add, multiply and fetch from main memory, are pipelined operations. This means that they are segmented into three stages, and in most cases a new operation can be initiated every 25 nanosecond clock period. Another advanced feature is the fact that multiple instructions can be executed in a single clock period. For example, a memory fetch, a floating add and a floating multiply can all be initiated in a single clock period.

A single node of the Touchstone Gamma system consists of the i860, 8 megabytes (MB) of dynamic random access memory, and hardware for communication to other nodes. For every 16 nodes, there is also a unit service module to facilitate access to the nodes for diagnostic purposes. The Touchstone Gamma system at NASA Ames consists of 128 computational nodes. The theoretical peak performance of this system is thus approximately 7.5 GFLOPS on 64 bit data.

The 128 nodes are arranged in a seven dimensional hypercube using the direct connect routing module and the hypercube interconnect technology of the iPSC/2. The point to point aggregate bandwidth of the interconnect system, which is 2.8 MB/sec per channel, is the same as on the iPSC/2. However the latency for the message passing is reduced from about 350 microseconds to about 90 microseconds. This reduction is mainly obtained through the increased speed of the i860 on the Touchstone Gamma machine, when compared to the Intel 386/387 on the iPSC/2. The improved latency is thus mainly a product of faster execution of the message passing software on the i860.

Attached to the 128 computational nodes of the NASA Ames system are

ten I/O nodes, each of which can store approximately 700 MB. The total capacity of the I/O system is thus about 7 GB. These I/O nodes operate concurrently for high throughput rates. The complete system is controlled by a system resource module (SRM), which is based on an Intel 80386 processor. This system handles compilation and linking of source programs, as well as loading the executable code into the hypercube nodes and initiating execution. At present the SRM is a serious bottleneck in the system, due to its slowness in compiling and linking user codes. For example, the compilation of a moderate-sized application program often requires 30 minutes or more, even with no optimization options and no other users on the system.

During 1990 the iPSC/860 has been thoroughly investigated at NASA Ames. A first set of benchmark numbers, and some CFD applications performance numbers have been published in [2]. A more recent summary is given by Barszcz in [5]. As documented in [5] from an overall systems aspect the main bottleneck has been the SRM, which is not able to handle the demands of a moderately large user community (about 50 to 100 users) in a production environment. Another important result of the investigations was the outcome of a study by Lee [15]. Lee's analysis of the i860 floating point performance indicates that on typical CFD kernels the best performance to be expected is in the 10 MFLOPS range. Finally we mention a performance study of the I/O system by Lou [19], which measures the I/O performance of the CFS.

3 Structured Grid Applications

Structured grid codes, in particular multiblock structured grid codes, are one of the main production CFD tools at NASA Ames. A number of different efforts were directed toward the implementation of such capabilities on parallel machines. One of the first CFD results on the CM-2 was the work by Levit and Jespersen [17, 16], which was recently extended to three dimensions [18]. Their implementation is based on the successful ARC2D and ARC3D codes developed by Pulliam [24]. Work by Barszcz and Chawla [6] is in progress to implement F3D, a successor code to ARC3D, on the CM-2. On the iPSC/860 Weeratunga has implemented ARC2D (for early results see [2]), and work is in progress to implement F3D. Weeratunga also has developed a pseudo CFD application based on structured grids for the NAS Parallel Benchmark, which

is described in chapter 3 of [4]. We will not discuss these efforts here in more detail and refer the interested reader to the references.

4 Unstructured Grid Applications

We discuss here work on an upwind finite-volume flow solver for the Euler equations in two dimensions that is well suited for massively parallel implementation. The mathematical formulation of this flow solver was proposed and implemented on the Cray-2 by Barth and Jespersen[7]. This solver has been implemented on the CM-2 by Hammond and Barth [12], and on the Intel iPSC/860 by Venkatakrishnan, Simon, and Barth [27].

The unstructured grid code developed by Barth is a vertex-based finite volume scheme. The control volumes are non-overlapping polygons which surround the vertices of the mesh, called the "dual" of the mesh. Associated with each edge of the original mesh is a dual edge. Fluxes are computed along each edge of the dual in an upwind fashion using an approximate Riemann solver. Piecewise linear reconstruction is employed which yields second order accuracy in smooth regions. A 4 stage Runge-Kutta scheme is used to advance the solution in time. Fluxes, gradients and control volumes are all constructed by looping over the edges of the original mesh. In the Cray implementation, vectorization is achieved by coloring the edges of the mesh.

It is assumed that a triangularization of the computational domain and the corresponding mesh has been computed. We will not present any more details here. A complete description of the algorithm can be found in [7, 12].

In both implementations the same test case has been used. The test case used is an unstructured mesh with 15606 vertices, 45878 edges, 30269 faces, 4 bodies, and 949 boundary edges. The flow was computed at a Mach number of .1 at 0 degrees angle of attack. The code for this test case runs at 150 Mflops on the NAS Cray-YMP at NASA Ames, and requires 0.39 seconds per time step.

4.1 SIMD Implementation of Unstructured Solver

For the implementation on the CM-2 Hammond and Barth [12] used a novel partitioning of the problem which minimizes the computation and commu-

nication costs on a massively parallel computer. In a mesh-vertex scheme, solution variables are associated with each vertex of the mesh and flux computation is performed at edges of the non-overlapping control volumes which surround each vertex. In conventional parallel implementations this operation is partitioned to be performed edge-wise, i.e., each *edge* of the control volume is assigned to one processor (edge-based). The resulting flux calculation contributes to two control volumes which share the particular edge.

In the partitioning used by Hammond and Barth, each *vertex* of the mesh is assigned to one processor (vertex-based). Flux computations are identical to the edge-based scheme but computed by processors associated with vertices. Each edge of the mesh joins a pair of vertices and is associated with one edge of the control volume.

One can direct edge (i,j) to determine which vertex in the pair computes the flux through the shared edge of the control volume, (k', j'). When there is a directed edge from i to j, then the processor holding vertex j sends its conserved values to the processor holding vertex i, and the flux across the common control volume edge is computed by processor i and accumulated locally. The flux through (k', j') computed by the processor holding vertex i is sent to the processor holding vertex j to be accumulated negatively. Hammond and Barth show that their vertex-based scheme requires 50% less communication and asymptotically identical amounts of computation as compared with the traditional edge-based approach.

Another important feature of the work by Hammond and Barth is the use of fast communication. A feature of the communication within the flow solver here is that the communication pattern, although irregular, remains static throughout the duration of the computation. The SIMD implementation takes advantage of this by using a mapping technique developed by Hammond and Schreiber [13] and a "Communication Compiler" developed for the CM-2 by Dahl [11]. The former is a highly parallel graph mapping algorithm that assigns vertices of the grid to processors in the computer such that the sum of the distances that messages travel is minimized. The latter is a software facility for scheduling irregular communications with a static pattern. The user specifies a list of source locations and destinations for messages which are then compiled into routing paths to be used at run time.

Hammond and Barth have incorporated the mapping algorithm and the communication compiler into the flow solver running on the CM-2 and have realized a factor of 30 reduction in communication time compared to using

naive or random assignments of vertices to processors and the router. Using 8K processors of the CM-2 and a VP ratio of 2, Hammond and Barth carried out 100 time steps of the flow solver in about 71.62 seconds. This does not include setup time.

4.2 MIMD Implementation of Unstructured Solver

Similar to the SIMD implementation one of the key issues is the partitioning of the unstructured mesh. In order to partition the mesh Venkatakrishnan et al. [27] employ a new algorithm for the graph partitioning problem, which has been discussed recently by Simon [26], and which is based on the computation of eigenvectors of the Laplacian matrix of a graph associated with the mesh. Details on the theoretical foundations of this strategy can be found in [23]. Detailed investigations and comparisons to other strategies (cf. [26]) have shown that the spectral partitioning produces subdomains with the shortest boundary, and hence tends to minimize communication cost.

After the application of the partition algorithm of the previous section, the whole finite volume grid with triangular cells is partitioned into P subgrids, each subgrid contains a number of triangular cells which form a single connected region. Each subgrid is assigned to one processor. All connectivity information is precomputed, using sparse matrix type data structures.

Neighboring subgrids communicate to each other only through their interior boundary vertices which are shared by the processors containing the neighboring subgrids. In the serial version of the scheme, field quantities (mass, momentum and energy) are initialized and updated at each vertex of the triangular grid using the conservation law for the Euler equations applied to the dual cells. Each processor performs the same calculations on each subgrid as it would do on the whole grid in the case of a serial computation. The difference is that now each subgrid may contain both physical boundary edges and interior boundary edges, which have resulted from grid partitioning. Since a finite volume approach is adopted, the communication at the inter-processor boundaries consists of summing the local contributions to integrals such as volumes, fluxes, gradients etc.

The performance of the Intel iPSC/860 on the test problem is given in Table 1.

Table 1: **Performance of Unstructured Grid Code on the Intel iPSC/860**

Processors	secs/step	MFLOPS	efficiency(%)
2	7.58	7.7	83
4	3.82	15.3	83
8	2.01	29.1	79
16	1.11	52.7	71
32	0.61	95.9	65
64	0.33	177.3	60
128	0.21	278.6	47

5 Particle Methods

Particle methods of simulation are of interest primarily for high altitude, low density flows. When a gas becomes sufficiently rarefied the constitutive relations of the Navier-Stokes equations (i.e. the Stokes law for viscosity and the Fourier law for heat conduction) no longer apply and either higher order relations must be employed (e.g. the Burnett equations [20]), or the continuum approach must be abandoned and the molecular nature of the gas must be addressed explicitly. The latter approach leads to direct particle simulation.

In direct particle simulation, a gas is described by a collection of simulated molecules thus completely avoiding any need for differential equations explicitly describing the flow. By accurately modelling the microscopic state of the gas the macroscopic description is obtained through the appropriate integration. The primary disadvantage of this approach is that the computational cost is relatively large. Therefore, although the molecular description of a gas is accurate at all densities, a direct particle simulation is competitive only for low densities where accurate continuum descriptions are difficult to make.

For a small discrete time step, the molecular motion and collision terms of the Boltzmann equation may be decoupled. This allows the simulated particle flow to be considered in terms of two consecutive but distinct events in one time step, specifically there is a collisionless motion of all particles followed by a motionless collision of those pairs of particles which have been identified as colliding partners. The collisionless motion of particles is strictly

deterministic and reversible. However, the collision of particles is treated on a probabilistic basis. The particles move through a grid of cells which serves to define the geometry, to identify colliding partners, and to sample the macroscopic quantities used to generate a solution.

The state of the system is updated on a per time step basis. A single time step is comprised of five events:

1. Collisionless motion of particles.
2. Enforcement of boundary conditions.
3. Pairing of collision partners.
4. Collision of selected collision partners.
5. Sampling for macroscopic flow quantities.

Detailed description of these algorithms may be found in [21] and [8]

5.1 SIMD Implementation of Particle Simulation

Particle simulation is distinct from other CFD applications in that there are two levels of parallel granularity in the method. There is a coarse level consisting of cells in the simulation (which are approximately equivalent to grid points in a continuum approach) and there is a fine level consisting of individual particles. At the time of the CM-2 implementation there existed only the fieldwise model of the machine, and it was natural for Dagum [8] to decompose the problem at the finest level of granularity. In this decomposition, the data for each particle is stored in an individual virtual processor in the machine. A separate set of virtual processors (or VP set) stores the geometry and yet another set of virtual processors stores the sampled macroscopic quantities.

This decomposition is conceptually pleasing however in practice the relative slowness of the Connection Machine router can prove to be a bottleneck in the application. Dagum [8] introduces several novel algorithms to minimize the amount of communication and improve the overall performance in such a decomposition. In particular, steps 2 and 3 of the particle simulation algorithm require a somewhat less than straightforward approach.

The enforcement of boundary conditions requires particles which are about to interact with a boundary to get the appropriate boundary information from the VP set storing the geometry data. Since the number of particles undergoing boundary interaction is relatively small, a master/slave algorithm is used to minimize both communication and computation. In this algorithm, the master is the VP set storing the particle data. The master creates a slave VP set large enough to accommodate all the particles which must undergo boundary interactions. Since the slave is much smaller than the master, instructions on the slave VP set execute much faster. This more than makes up for the time that the slave requires to get the geometry information and to both get and return the particle information.

The pairing of collision partners requires sorting the particle data such that particles occupying the same cell are represented by neighboring virtual processors in the one dimensional NEWS grid storing this data. Dagum [9] describes different sorting algorithms suitable for this purpose. The fastest of these makes use of the realization that the particle data moves through the CM processors in a manner analogous to the motion of the particles in the simulation. The mechanism for disorder is the motion of particles, and the extent of motion of particles, over a single time step, is small. This can be used to tremendously reduce the amount of communication necessary to re-order the particles.

These algorithms have been implemented in a two-dimensional particle simulation running on the CM-2. At the time of implementation, the CM-2 at NASA Ames had only 64k bits of memory per processor which was insufficient to warrant a three-dimensional implementation. Furthermore, the slicewise model of the machine did not exist and the machine had the slower 32-bit Weitek's which did not carry out any integer arithmetic. Nonetheless, with this smaller amount of memory and fieldwise implementation, the code was capable of simulating over 2.0×10^6 particles in a grid with 6.0×10^4 at a rate of 2.0μsec/particle/timestep using all 32k processors (see [8]). By comparison, a fully vectorized equivalent simulation on a single processor of the Cray YMP runs at 1.0μsec/particle/timestep and 86 MFLOPS as measured by the Cray hardware performance monitor. (Note that a significant fraction of a particle simulation involves integer arithmetic and the MFLOP measure is not completely indicative of the amount of computation involved). Currently, work is being carried out to extend the simulation to three dimensions using a parallel decomposition which takes full advantage

of the slicewise model of the machine.

5.2 MIMD Implementation of Particle Simulation

The MIMD implementation differs from the SIMD implementation not so much because of the difference in programming models but because of the difference in granularity between the machine models. Whereas the CM-2 has 32768 processors, the iPSC/860 has only 128. Therefore on the iPSC/860 it is natural to apply a spatial domain decomposition rather than the data object decomposition used on the CM-2.

In McDonald's [22] implementation, the spatial domain of the simulation is divided into a number of sub-domains or regions equal to the desired number of node processes. Communication between processes occurs as a particle passes from one region to another and is carried out asynchronously, thus allowing overlapping communication and computation. Particles crossing region "seams" are treated simply as an additional type of boundary condition. Each simulated region of space is surrounded by a shell of extra cells that, when entered by a particle, directs that particle to the neighboring region. This allows the representation of simulated space (i.e. the geometry definition) to be distributed along with the particles. The aim is to avoid maintaining a representation of all simulated space which, if stored on a single processor, would quickly become a serious bottleneck for large simulations, and if replicated would simply be too wasteful of memory.

Within each region the sequential or vectorized particle simulation is applied. This decomposition allows for great flexibility in the physical models that are implemented since node processes are asynchronous and largely independent of each other. Recall that communication between processes is required only when particles cross region seams. This is very fortuitous since the particle motion is straightforward and fully agreed upon. The important area of research has to do with the modelling of particles, and since this part of the problem does not directly affect communication, particle models can evolve without requiring great algorithmic changes.

McDonald's implementation is fully three-dimensional. The performance of the code on a 3D heat bath is given in Table 2.

At the present time the domain decomposition is static, however work is being carried out to allow dynamic domain decomposition thus permitting a good load balance to exist throughout a calculation. The geometry and

Table 2: **Performance of Particle Simulation on the Intel iPSC/860**

Processors	μs/prt/step	MFLOPS	efficiency(%)
2	24.4	3.5	97
4	12.5	6.9	95
8	6.35	13.5	93
16	3.25	26.5	91
32	1.63	52.8	91
64	0.85	101	87
128	0.42	215	88

spatial decomposition of the heat bath simulation *exaggerated* the area to volume ratio of the regions in order to more closely approximate the communication expected in a real application with dynamic load balancing. The most promising feature of these results is the linear speed up obtained, indicating that the performance of the code should continue to increase with increasing numbers of processors.

6 Conclusions

On the unstructured grid code the performance figures are summarized in Table 3, where all MFLOPS numbers are Cray Y-MP equivalent numbers.

Table 3: **Performance Comparison of Unstructured Grid Code**

Machine	Processors	secs/step	MFLOPS
Cray Y-MP	1	0.39	150.0
Intel iPSC/860	64	0.33	177.3
	128	0.21	278.6
CM-2 (32 bit)	8192	0.72	81.3

For the particle methods the corresponding summary of performance figures can be found in Table 4. The figures in Table 4 should be interpreted very carefully. The simulations run on the different machines were comparable, but not identical. The MFLOPS are Cray Y-MP equivalent MFLOPS ratings based on the hardware performance monitor.

Table 4: **Performance Comparison of Particle Simulation Code**

Machine	Processors	μsecs/particle/step	MFLOPS
Cray 2	1	2.0	43
Cray Y-MP	1	1.0	86
Intel iPSC/860	128	0.4	215
CM-2 (32 bit)	32768	2.0	43

The results in Tables 3 and 4 demonstrate a number of points. Both unstructured grid computations and the particle simulations are applications which a priori are not immediately parallelized, and for which both on SIMD and MIMD machines considerable effort must be expended in order to obtain an efficient implementation. It has been demonstrated by the results obtained at NASA Ames that this can be done, and that supercomputer level performance can be obtained on current generation parallel machines. Furthermore the particle simulation code on the CM-2 is a production code currently used to obtain production results (see [10]). The iPSC/860 implementation should be in production use by the end of 1991.

Our results also demonstrate another feature which has been found across a number of applications at NASA Ames: massively parallel machines quite often obtain only a fraction of their peak performance on realistic applications. In the applications considered here, the requirement for unstructured, general communication has been the primary impediment in obtaining the peak realizable performance from these machines. Neither the CM-2 nor the the iPSC/860 deliver the communication bandwidth necessary for these CFD applications. This situation is even worse for implicit algorithms (see e.g. [6, 2]). Experience has shown that CFD applications require on the order of one memory reference per floating point operation and a balanced system should have a memory bandwidth comparable to its floating point performance. In these terms, current parallel systems deliver only a fraction of the required bandwidth.

Acknowledgement. We wish to thank our colleagues with the NAS Applied Research Branch, whose work has been discussed here: D. Bailey, E. Barszcz, R. Fatoohi, T. Lasinski, C. Levit, V. Venkatakrishnan, and S. Weeratunga. We also thank T. Barth, D. Jespersen, J. McDonald, (all NASA Ames) and P. Fredericksen, S. Hammond, and R. Schreiber at RIACS for

their contributions to this summary report.

References

[1] *Numerical Aerodynamic Simulation Program Plan.* NAS Systems Division, NASA Ames Research Center, October 1988.

[2] D. Bailey, E. Barszcz, R. Fatoohi, H. Simon, and S. Weeratunga. Performance results on the intel touchstone gamma prototype. In David W. Walker and Quentin F. Stout, editors, *Proceedings of the Fifth Distributed Memory Computing Conference*, pages 1236 – 1246, IEEE Computer Society Press, Los Alamitos, California, 1990.

[3] D. H. Bailey. *Experience with Parallel Computers at NASA Ames.* Technical Report RNR-91-07, NASA Ames Research Center, Moffett Field, CA 94035, February 1991.

[4] D. H. Bailey, J. Barton, T. Lasinski, and H. Simon. *The NAS Parallel Benchmarks.* Technical Report RNR-91-02, NASA Ames Research Center, Moffett Field, CA 94035, January 1991.

[5] E. Barszcz. *One Year With an iPSC/860.* Technical Report RNR-91-01, NASA Ames Research Center, Moffett Field, CA 94035, January 1991.

[6] E. Barszcz and K. Chawla. F3D On the CM-2. In T. Pulliam, editor, *Compendium of Abstracts, NASA CFD Conference, March 1991*, pages 56 – 57, NASA Office of Aeronautics Exploration and Technology, March 1991.

[7] T.J. Barth and D.C. Jespersen. The design and application of upwind schemes on unstructured meshes. In *Proceedings, 27th Aerospace Sciences Meeting*, January 1989. Paper AIAA 89-0366.

[8] L. Dagum. *On the Suitability of the Connection Machine for Direct Particle Simulation.* Technical Report 90.26, RIACS, NASA Ames Research Center, Moffett Field, CA 94035, June 1990.

[9] L. Dagum. Sorting for particle flow simulation on the connection machine. In Horst D. Simon, editor, *Research Directions in Parallel CFD*, MIT Press, Cambridge(to appear), 1991.

[10] L. Dagum. *Lip Leakage Flow Simulation for the Gravity Probe B Gas Spinup Using PSiCM.* Technical Report RNR-91-10, NASA Ames Research Center, Moffett Field, CA 94035, March 1991.

[11] E. Denning Dahl. Mapping and compiled communication on the connection machine system. In David W. Walker and Quentin F. Stout, editors, *Proceedings of the Fifth Distributed Memory Computing Conference*, pages 756 – 766, IEEE Computer Society Press, Los Alamitos, California, 1990.

[12] S. Hammond and T.J. Barth. On a massively parallel Euler solver for unstructured grids. In Horst D. Simon, editor, *Research Directions in Parallel CFD*, MIT Press, Cambridge(to appear), 1991.

[13] S. Hammond and R. Schreiber. *Mapping Unstructured Grid Problems to the Connection Machine.* Technical Report 90.22, RIACS, NASA Ames Research Center, Moffett Field, CA 94035, October 1990.

[14] Intel Corporation. *i860 64-Bit Microprocessor Programmer's Reference Manual.* Santa Clara, California, 1990.

[15] K. Lee. *On the Floating Point Performance of the i860 Microprocessor.* Technical Report RNR-90-019, NASA Ames Research Center, Moffett Field, CA 94035, 1990.

[16] C. Levit and D. Jespersen. A computational fluid dynamics algorithm on a massively parallel computer. *Int. J. Supercomputer Appl.*, 3(4):9 – 27, 1989.

[17] C. Levit and D. Jespersen. *Explicit and Implicit Solution of the Navier-Stokes Equations on a Massively Parallel Computer.* Technical Report, NASA Ames Research Center, Moffett Field, CA, 1988.

[18] C. Levit and D. Jespersen. *Numerical Simulation of a Flow Past A Tapered Cylinder.* Technical Report RNR-90-20, NASA Ames Research Center, Moffett Field, CA 94035, October 1990.

[19] Zhong C. Lou. *A Summary of CFS I/O Tests.* Technical Report RNR-90-20, NASA Ames Research Center, Moffett Field, CA 94035, October 1990.

[20] F.E. Lumpkin. *Development and Evaluation of Continuum Models for Translational-Rotational Nonequilibrium.* PhD thesis, Stanford University, Dept. of Aeronautics and Astronautics, Stanford CA 94305, April 1990.

[21] J. D. McDonald. *A Computationally Efficient Particle Simulation Method Suited to Vector Computer Architectures.* PhD thesis, Stanford University, Dept. of Aeronautics and Astronautics, Stanford CA 94305, December 1989.

[22] J. D. McDonald. *Particle Simulation in a Multiprocessor Environment.* Technical Report RNR-91-02, NASA Ames Research Center, Moffett Field, CA 94035, January 1991.

[23] A. Pothen, H. Simon, and K.-P. Liou. Partitioning sparse matrices with eigenvectors of graphs. *SIAM J. Mat. Anal. Appl.*, 11(3):430 – 452, 1990.

[24] T. H. Pulliam. Efficient solution methods for the Navier-Stokes equations. 1986. Lecture Notes for The Von Karman Institute for Fluid Dynamics Lecture Series, Jan. 20 - 24.

[25] R. Schreiber. *An Assessment of the Connection Machine.* Technical Report 90.40, RIACS, NASA Ames Research Center, Moffett Field, CA 94035, June 1990.

[26] H. D. Simon. *Partitioning of Unstructured Problems for Parallel Processsing.* Technical Report RNR-91-08, NASA Ames Research Center, Moffett Field, CA 94035, February 1991. (to appear in Computing Systems in Engineering).

[27] V. Venkatakrishnan, H. Simon, and T. Barth. *A MIMD Implementation of a Parallel Euler Solver for Unstructured Grids.* Technical Report RNR-91-xx, NASA Ames Research Center, Moffett Field, CA 94035, 1991. (in preparation).